Jiten Gurung

CMPS 480

Prof. Mark Voortman

API Documentation

App.js serves as the main entry point for a Node.js application that utilizes the Express framework to provide a RESTful API. This application is connected to a PostgreSQL database and is configured to handle CORS (Cross-Origin Resource Sharing) to allow requests from different origins. It also parses JSON and URL-encoded data from incoming requests. Below is a comprehensive guide to understanding and interacting with this application.

Application Setup

This section describes the various modules and middlewares integrated into the application.

Modules and Middleware

Module	Purpose
Express:	Framework to build the server and manage routes
Cors:	Middleware to enable CORS
Postgresql:	PostgreSQL client for connecting to PostgreSQL databases
Body-parser:	Middleware to parse JSON and URL-encoded data in requests

```
const express = require('express');
const cors = require('cors');
const bodyParser = require('body-parser');
const app = express();

app.use(cors());
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));
```

Database Configuration

The application connects to a PostgreSQL database using the pg module. Database credentials and details are specified directly in the code.

PostgreSQL Pool Setup

```
const pool = new pg.Pool({
   user: 'skillsculpt',
   host: ‘remote’,  (hiding the actual remote hostname)
   database: 'skillsculpt',
   password: 'password', (hiding the actual password)
   port: 5432,
   ssl: { rejectUnauthorized: false }
});
```

API Endpoints

GET /api/data

Retrieves all entries from the employees table in the database.

Endpoint: **GET /api/data**

Functionality:

- Fetches all employee data.
- Returns data in JSON format.

POST /api/newentry

Allows adding a new entry to the employees table.

Endpoint: **POST /api/newentry**

Payload Example:

```
{
   "first_name": "Alex",
   "last_name": "Smith",
   "position": "Developer",
   "id": 123
}
```

Functionality:

- Inserts new employee data into the database.
- Returns a success message upon successful insertion.

DELETE /api/delete/:id

Deletes an employee entry based on the provided ID.

Endpoint: **DELETE** **/api/delete/:id**

Functionality:

- Deletes the employee with the specified ID.
- Returns a success message upon successful deletion.

Server Initialization

The server listens on port 3000 and prints a message to the console once it starts successfully.

```
const port = 3000;
app.listen(port, () => {
   console.log(`Server is running on port ${port}`);
});
```

This documentation provides a clear overview of the functionalities provided by the application, including how it interfaces with a PostgreSQL database and handles various API requests.